IJIRSET
International Journal of Innovative Research in Science | Engineering | Technology

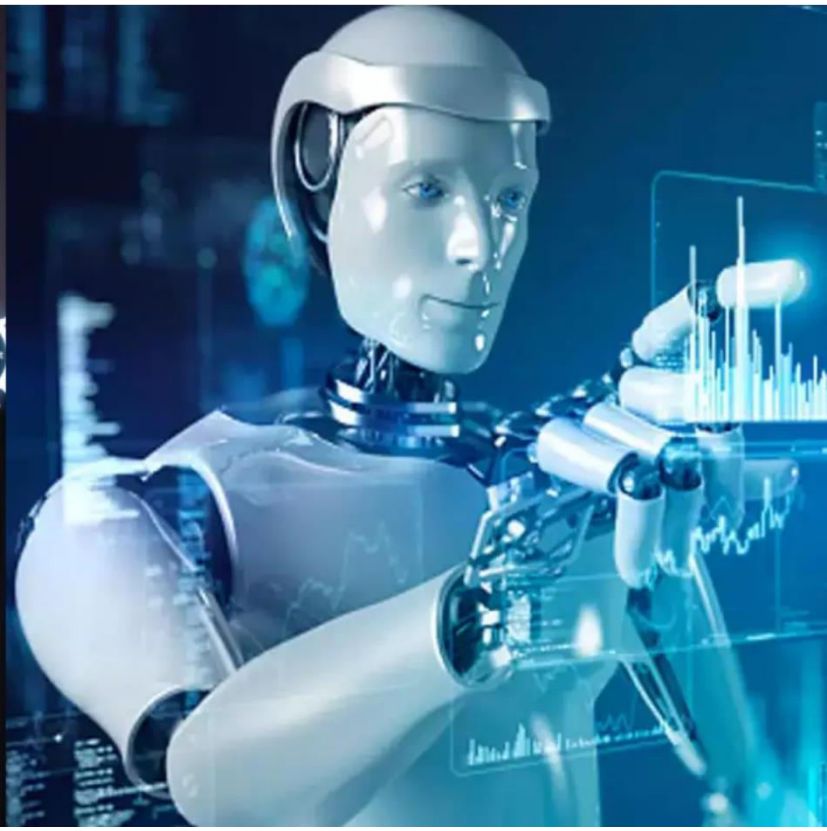
ISSN(O): 2319-8753
ISSN(P): 2347-6710



International Journal of Innovative Research in Science Engineering and Technology (IJIRSET)

(A Monthly, Peer Reviewed, Refereed, Scholarly Indexed, Open Access Journal)





Impact Factor: 9.935

Volume 15, Issue 11, November 2026

International Journal of Innovative Research of Science, Engineering and Technology (IJIRSET)

Impact Factor 9.935

|www.ijirset.com |A Monthly, Peer Reviewed & Refereed Journal| e-ISSN: 2319-8753| p-ISSN: 2347-6710|

Volume 15, Issue 11, November 2026

|DOI: 10.15680/IJIRSET.2026.1511045|

Privacy Shield: A Privacy-Preserving AI Interaction Platform

Aditya Mapari[1], Harshada Dighe[2], Swapnil Shirsath[3], Prof. Kale I.A[4]

Department of Artificial Intelligence and Data Science, Vishwabharati Academy's College of Engineering, Ahilyanagar, Maharashtra, India[1-3]

Guide, Department of Artificial Intelligence and Data Science, Vishwabharati Academy's College of Engineering, Ahilyanagar, Maharashtra, India[4]

ABSTRACT: The widespread adoption of external Large Language Models (LLMs) across academic, enterprise, and personal workflows has elevated the risk of inadvertent disclosure of Personally Identifiable Information (PII) and confidential organizational records inside user prompts. Once transmitted beyond a user's trusted application boundary, external providers retain varying degrees of visibility, logging, and data-retention control over raw prompt text. Existing manual redaction is error-prone, whereas irreversible anonymization frequently strips semantic and structural cues required for accurate model reasoning. This paper presents the system architecture, processing methodology, security model, and evaluation framework of Privacy Shield, a privacy-preserving middleware platform positioned between end users and external LLM APIs. The proposed multi-tier architecture decouples user interaction, RESTful orchestration, and natural language privacy analysis across a React.js frontend, a Java Spring Boot orchestration backend, and a Python FastAPI privacy-analysis microservice leveraging the Microsoft Presidio engine. By combining pattern-based recognizers with Named Entity Recognition (NER), the middleware detects sensitive spans, maps extracted values to session-scoped temporary placeholders within a trusted token store, transmits only the sanitized prompt to external inference endpoints, and performs schema-validated restoration of permitted placeholders prior to client display. To maintain strict scientific rigor, this manuscript explicitly distinguishes currently prototyped detection and tokenization workflows from planned production features—including live external LLM API integration, encrypted persistent token storage, and optical character recognition (OCR)—and defines a reproducible, multi-dimensional evaluation protocol to benchmark entity detection, leakage prevention, restoration fidelity, and latency overhead.

KEYWORDS: Large Language Models, Personally Identifiable Information, Privacy-Preserving Middleware, Prompt Sanitization, Reversible Tokenization, Named Entity Recognition, Microsoft Presidio, Trust Boundary.

I. INTRODUCTION

Large Language Models (LLMs) have become foundational tools for text generation, document summarization, software engineering, and analytical decision support. In routine workflows, users frequently embed sensitive attributes—such as personal names, electronic mail addresses, telephone numbers, residential locations, financial account numbers, and government-issued identifiers—directly into natural-language prompts. When prompts are dispatched over public networks to third-party inference APIs, users relinquish direct governance over how those payloads are logged, cached, audited, or incorporated into downstream model refinement pipelines [1]. Consequently, organizations and individuals face a persistent tension between leveraging cloud-hosted generative AI capabilities and complying with data-minimization principles.

Current practitioner workarounds exhibit substantial operational limitations. Manual prompt redaction relies entirely on human vigilance, making it slow, inconsistent, and vulnerable to oversight when processing lengthy or multi-turn inputs. Conversely, naive or irreversible anonymization—such as replacing all detected entities with generic asterisks or static redaction blocks—eliminates sensitive exposure but simultaneously degrades referential continuity, preventing the LLM from distinguishing between multiple actors, dates, or accounts within the same prompt [2]. Effective privacy mediation therefore requires a mechanism that isolates raw sensitive values within a user-controlled perimeter while preserving structural and contextual cues needed for coherent model output.

International Journal of Innovative Research of Science, Engineering and Technology (IJIRSET)

Impact Factor 9.935

|www.ijirset.com |A Monthly, Peer Reviewed & Refereed Journal| e-ISSN: 2319-8753| p-ISSN: 2347-6710|

Volume 15, Issue 11, November 2026

|DOI: 10.15680/IJIRSET.2026.1511045|

To address this gap, this paper presents Privacy Shield, a modular privacy-preserving middleware platform designed to intercept, sanitize, and reconstruct LLM interactions. Rather than modifying underlying foundation model weights or requiring local hosting of resource-intensive LLMs, Privacy Shield operates as an intermediary service layer. The architecture separates concerns across a React.js client interface, a Java Spring Boot REST API orchestrator, and a dedicated Python FastAPI privacy-analysis microservice built upon Microsoft Presidio [3]. Sensitive entities identified through hybrid regular-expression (Regex) and Named Entity Recognition (NER) pipelines are substituted with deterministic, structured temporary placeholders (e.g., <PERSON_1>, <EMAIL_1>) backed by an isolated token-mapping store before any external API transmission occurs.

The principal contributions of this work are summarized as follows:

- **Decoupled Multi-Service Privacy Architecture:** We specify a clear separation of responsibilities between user-interface rendering (React.js), request lifecycle orchestration and policy enforcement (Java Spring Boot), and statistical/rule-based entity recognition (Python FastAPI wrapping Microsoft Presidio).
- **Context-Preserving Reversible Sanitization Workflow:** We formulate a nine-stage processing pipeline that pairs entity classification and span tracking with structured placeholder substitution, enabling downstream LLMs to reason over entity relationships without accessing underlying PII values.
- **Trust-Boundary Security Model and Validation Controls:** We define explicit trust boundaries governing token scoping, outbound payload verification, and strict placeholder validation upon model response ingestion to prevent malformed-token injection or cross-session leakage.
- **Transparent Prototype Framing and Evaluation Protocol:** We establish a clear engineering distinction between implemented prototype components and planned production extensions, accompanied by a rigorous, multi-metric evaluation methodology ready for empirical benchmarking.

Importantly, Privacy Shield is positioned as a risk-reduction layer rather than an absolute cryptographic guarantee against all forms of privacy leakage. Overall protection depends on entity detection recall, contextual inference resistance, strict enforcement of the application trust boundary, and the operational security of the token store.

II. BACKGROUND AND RELATED WORK

Privacy risks in third-party LLM usage stem from both direct memorization/logging of explicit identifiers and indirect inference from contextual combinations. In standard cloud inference deployments, the client transmits raw UTF-8 prompt strings directly to the model provider's endpoint. Even where providers offer zero-data-retention agreements, regulatory frameworks and institutional security policies often prohibit transmitting unredacted PII outside controlled enterprise boundaries.

Research on privacy-preserving LLM interaction broadly spans three technical paradigms: (1) model-side cryptographic or differential privacy techniques, such as homomorphic encryption and differentially private fine-tuning, which incur high computational overhead or require provider-side integration; (2) local on-premise model deployment, which eliminates external transmission at the cost of substantial hardware infrastructure and lower reasoning capability compared to frontier cloud models; and (3) client-side or middleware prompt sanitization, which transforms sensitive text prior to external transmission while preserving compatibility with commercial black-box APIs [1], [2].

Within the prompt-sanitization paradigm, hybrid PII detection frameworks have emerged as practical building blocks. Rule-based pattern recognizers using regular expressions and checksum validation achieve high precision on structured formats such as email addresses, IP addresses, and national identification numbers, whereas statistical and neural NER models capture unstructured entities such as personal names, organizations, and geographic locations [3]. Modern privacy toolkits—such as Microsoft Presidio [3] and LLM Guard [4]—combine these recognizers with reversible anonymization or vault-backed deanonymization workflows. However, integrating Python-native NLP sanitization libraries into enterprise web ecosystems commonly built on Java Spring Boot and React.js requires clean service boundaries, deterministic placeholder formatting that survives LLM generation intact, and rigorous response-validation controls.

International Journal of Innovative Research of Science, Engineering and Technology (IJIRSET)

Impact Factor 9.935

|www.ijirset.com |A Monthly, Peer Reviewed & Refereed Journal| e-ISSN: 2319-8753| p-ISSN: 2347-6710|

Volume 15, Issue 11, November 2026

|DOI: 10.15680/IJIRSET.2026.1511045|

III. LITERATURE SURVEY

A targeted review of recent literature in privacy-preserving prompt engineering, utility-aware text sanitization, and open-source PII protection toolkits informs the architectural design of Privacy Shield. Edemacu and Wu [1] provide a comprehensive survey of privacy-preserving prompt engineering, categorizing threat surfaces across user-LLM interactions and analyzing the structural trade-off between privacy guarantees and task utility. Their taxonomy highlights prompt sanitization and intermediate masking gateways as practical mechanisms for black-box API settings where users cannot alter model parameters.

To balance privacy protection against semantic degradation, Shen et al. [2] introduce ProSan, a utility-based prompt privacy sanitizer that evaluates token sensitivity alongside downstream task relevance rather than applying uniform redaction. Similarly, privacy-preserving text sanitization approaches—such as Privacy-Preserving Text Sanitization (PP-TS) frameworks explored in natural language processing privacy workshops [5]—demonstrate that structured entity replacement preserves syntactic roles and coreference chains far better than coarse removal. On the applied systems side, Microsoft Presidio [3] establishes a modular, extensible architecture separating entity analysis (Presidio Analyzer) from text transformation (Presidio Anonymizer), while Protect AI's LLM Guard [4] demonstrates an operational Anonymize/Vault/Deanonymize pattern specifically tailored for LLM prompt and response pipelines.
Table I summarizes the core focus, strengths, and identified limitations of these foundational works and toolkits relative to the objectives of Privacy Shield.

TABLE I. SUMMARY OF LITERATURE SURVEY AND RELATED SANITIZATION FRAMEWORKS

Reference / System	Primary Approach	Key Strengths	Identified Limitations / Gap Addressed
Edemacu & Wu (2025) [1] (Prompt Engineering Privacy Survey)	Taxonomy of prompt-level privacy threats and defenses for black-box LLMs.	Clarifies privacy-utility trade-off and supports sanitization gateways.	Conceptual taxonomy rather than a full-stack middleware architecture.
Shen et al. (2026) [2] (ProSan Prompt Privacy Sanitizer)	Utility-aware sensitivity scoring and selective prompt sanitization.	Preserves task performance while masking high-risk sensitive tokens.	Requires specialized optimization; lacks multi-tier REST & reversible UI flow.
Microsoft Presidio [3] (PII Detection & Anonymization SDK)	Python SDK combining Regex, checksum validators, and NER models.	Modular entity recognition with confidence scores and span offsets.	Native Python library; requires REST wrapping for Java/React web stacks.
Protect AI LLM Guard [4] (Anonymize / Deanonymize Scanners)	Scanner pipeline using a stateful vault to mask prompts and restore outputs.	Demonstrates end-to-end reversible tokenization for LLM interactions.	Needs strict schema validation and session scoping in custom enterprise tiers.
PP-TS Literature [5] (Text Sanitization Benchmarks)	Entity-level obfuscation and placeholder substitution in NLP privacy.	Typed placeholders maintain grammatical and coreferential coherence.	Contextual leakage and restoration drift require post-inference validation.

IV. PROPOSED SYSTEM ARCHITECTURE

Privacy Shield is architected as a decoupled, three-tier middleware system positioned between the end user's browser and external LLM inference providers. A critical architectural design decision is the separation between enterprise API orchestration and Python-based natural language processing. Because Microsoft Presidio is a native Python framework rather than a native Java library, embedding it directly inside a Java Virtual Machine (JVM) backend would introduce

International Journal of Innovative Research of Science, Engineering and Technology (IJIRSET)

Impact Factor 9.935

|www.ijirset.com |A Monthly, Peer Reviewed & Refereed Journal| e-ISSN: 2319-8753| p-ISSN: 2347-6710|

Volume 15, Issue 11, November 2026

|DOI: 10.15680/IJIRSET.2026.1511045|

fragile foreign-function bindings and dependency conflicts. Instead, Privacy Shield isolates privacy analysis inside a dedicated Python FastAPI microservice invoked over an internal REST interface by the Java Spring Boot orchestrator. The system comprises six primary logical components operating across the trusted and untrusted zones:

- **1) Presentation Layer (React.js Frontend):** Provides an interactive web interface where users compose prompts, inspect detected entity classifications (in transparency/preview modes), and view restored model responses.
- **2) Orchestration & Policy Layer (Java Spring Boot Backend):** Serves as the central REST API gateway. It authenticates client sessions, coordinates calls to the internal privacy microservice, manages session-scoped token persistence, enforces outbound payload checks, and routes sanitized requests to external AI services.
- **3) Privacy-Analysis Microservice (Python FastAPI):** Exposes low-latency internal HTTP endpoints that wrap Microsoft Presidio Analyzer and custom entity-mapping logic, returning detected entity spans, confidence scores, and canonical entity types to the Spring Boot backend.
- **4) Hybrid Detection Engine (Microsoft Presidio with NER & Regex):** Combines deterministic pattern recognizers (regular expressions and checksum rules for structured identifiers like emails, phone numbers, and ID codes) with statistical/transformer-based NER models for contextual entities such as person names and locations.
- **5) Token-Mapping Store (Trusted State Repository):** Maintains the bidirectional association between generated temporary placeholders (e.g., <PERSON_1>) and original raw values strictly within the trusted backend boundary.
- **6) External LLM Gateway & Response Validator:** Transmits strictly sanitized prompt text to the external LLM API and inspects returned completions to validate placeholder syntax before controlled re-substitution.

Table II details the technology stack and architectural role of each layer, explicitly distinguishing core prototype components from planned integration capabilities.

TABLE II. PRIVACY SHIELD TECHNOLOGY STACK AND COMPONENT ROLES

Architectural Layer	Technology / Framework	Primary Functional Role	Implementation Status
Client Interface	React.js (JS / HTML5 / CSS3)	Prompt submission UI, sanitized preview, and restored output view.	Prototype UI implemented
Backend Orchestrator	Java Spring Boot (REST API)	Request routing, workflow coordination, masking, and response validation.	Core REST orchestration prototyped
Privacy Microservice	Python FastAPI	Internal REST wrapper exposing Presidio detection to Spring Boot.	Prototype service implemented
PII Detection Engine	Microsoft Presidio Analyzer	Entity span identification, confidence scoring, and entity classification.	Implemented in prototype
Rule & NER Recognizers	NER Models + Regex Recognizers	Contextual detection (names, locations) + pattern matching (emails, IDs).	Implemented via Presidio recognizers
Sanitization & Mapping	Placeholders & Token Store	Deterministic tokens (<TYPE_N>) and session-scoped reverse mapping.	Prototype store; encrypted vault planned
External Inference	External LLM REST API	Generates completions solely from sanitized prompts with placeholders.	Interface defined; live API planned

International Journal of Innovative Research of Science, Engineering and Technology (IJIRSET)

Impact Factor 9.935

|www.ijirset.com |A Monthly, Peer Reviewed & Refereed Journal| e-ISSN: 2319-8753| p-ISSN: 2347-6710|

Volume 15, Issue 11, November 2026

|DOI: 10.15680/IJIRSET.2026.1511045|

Fig. 1 illustrates the end-to-end component topology and trust-boundary separation across the client, middleware services, and external LLM provider.

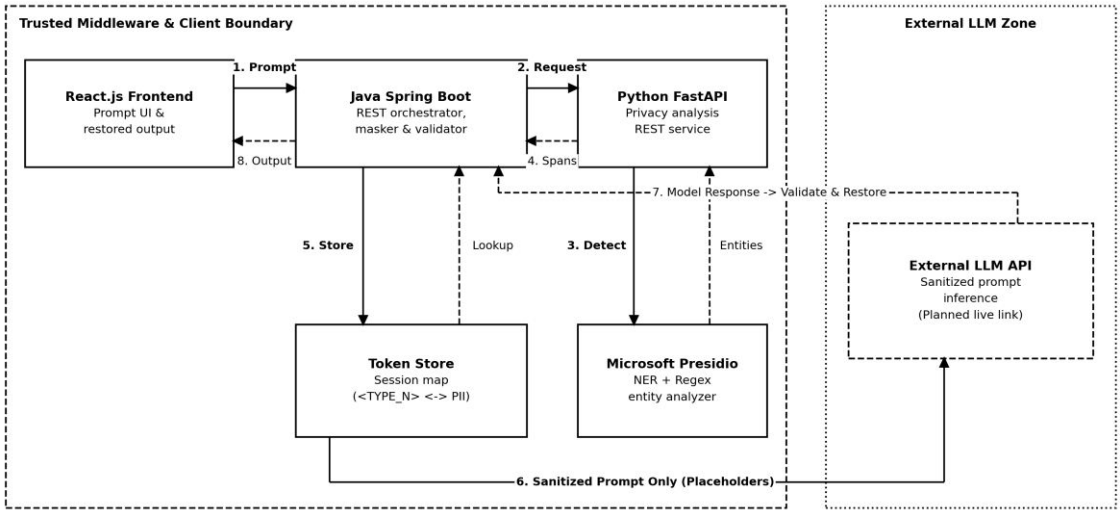


Fig. 1. Privacy Shield System Architecture illustrating component decoupling across the trusted middleware and client boundary (React.js, Java Spring Boot, Python FastAPI, Microsoft Presidio, and Token Store) and the external LLM zone.

V. METHODOLOGY AND IMPLEMENTATION

A. End-to-End Sanitization and Restoration Workflow

Privacy Shield executes a deterministic nine-step processing pipeline for each user prompt submission, depicted in Fig. 2:

- **Step 1 — Prompt Submission:** The user enters a natural-language prompt in the React.js interface, which transmits the payload over HTTPS to the Spring Boot backend along with a session identifier.
- **Step 2 — PII Detection:** Spring Boot forwards the raw prompt text to the Python FastAPI microservice, which invokes Microsoft Presidio Analyzer configured with both pattern-based recognizers (for emails, phone numbers, IP addresses, and structured IDs) and NER models (for person names, organizations, and locations).
- **Step 3 — Entity Classification and Span Resolution:** Presidio returns a list of candidate detections, each specifying the entity category, character start/end offsets, and confidence score. Overlapping spans are resolved using confidence and span-length precedence rules.
- **Step 4 — Placeholder Generation and Token Mapping:** For each confirmed sensitive span, the orchestrator checks whether an identical normalized value already exists within the active session map. Existing values reuse their assigned placeholder to preserve coreference; new values receive a sequential typed token (e.g., <PERSON_1>, <EMAIL_1>, <PHONE_1>). The mapping (placeholder → original value, entity type, offset metadata) is written to the trusted token store.
- **Step 5 — Sanitized Prompt Construction:** Original substrings are replaced in reverse offset order with their corresponding placeholders, yielding a sanitized prompt string free of detected PII while preserving sentence grammar and entity relationships.
- **Step 6 — External LLM Processing:** Only the sanitized prompt is dispatched to the external LLM API. System instructions appended to the request instruct the model to treat bracketed tokens such as <PERSON_1> as opaque entity identifiers and preserve them verbatim in its output.
- **Step 7 — Placeholder Validation:** Upon receiving the LLM completion, the Spring Boot response processor scans the generated text using a strict placeholder grammar parser. It verifies that every candidate token matches an active placeholder issued for the current request/session, rejecting malformed, hallucinated, or unauthorized token patterns.

International Journal of Innovative Research of Science, Engineering and Technology (IJIRSET)

Impact Factor 9.935

|www.ijirset.com |A Monthly, Peer Reviewed & Refereed Journal| e-ISSN: 2319-8753| p-ISSN: 2347-6710|

Volume 15, Issue 11, November 2026

|DOI: 10.15680/IJIRSET.2026.1511045|

- **Step 8 — Controlled Restoration:** Validated placeholders are replaced with their original values retrieved from the trusted token store according to configured disclosure policies (e.g., full restoration for the originating user or selective masking for shared views).
- **Step 9 — Response Display:** The reconstructed response is returned to the React.js frontend, optionally accompanied by metadata indicating which entity categories were protected during inference.

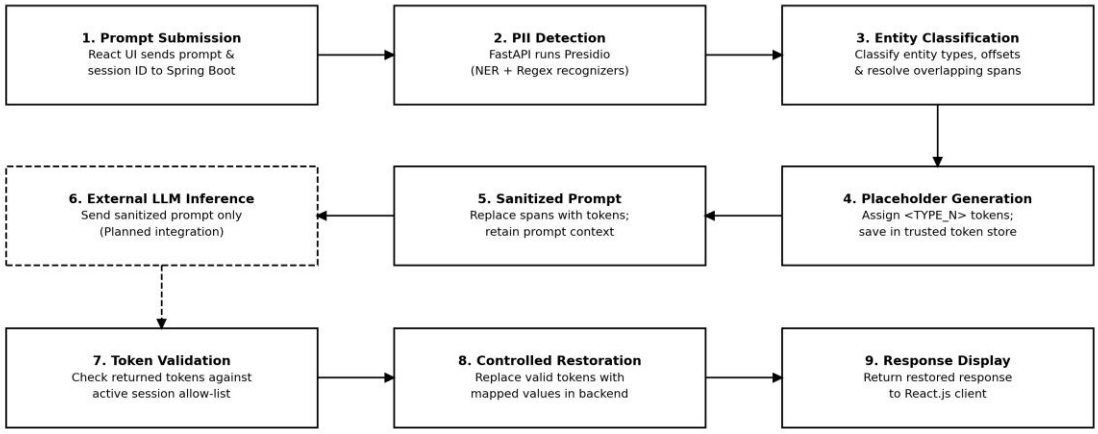


Fig.2. Privacy Shield Processing Workflow illustrating sequential execution from prompt submission through PII detection, placeholder tokenization, external inference (dashed block), validation, and controlled restoration.

B. Implementation Status: Prototype vs. Planned Extensions

To ensure transparent scientific reporting, Table III delineates the engineering maturity of each subsystem in the current implementation.

TABLE III. DISTINCTION BETWEEN PROTOTYPED COMPONENTS AND PLANNED SYSTEM CAPABILITIES

System Capability	Current Prototype Status	Planned / Future Production Enhancement
Frontend & Orchestration	React.js UI and Java Spring Boot REST endpoints implemented for local workflow execution.	Role-based authentication, entity-policy toggles, and audit dashboard.
PII Detection Pipeline	Python FastAPI service with Microsoft Presidio (Regex + NER recognizers) for English text.	Domain-specific recognizers, multilingual NER models, and threshold tuning.
Token-Mapping Store	Session-scoped prototype mapping structure supporting reversible placeholder lookup.	AES-256 encrypted persistent vault with TTL eviction and secure erasure.
External LLM Integration	Sanitized prompt construction verified locally; pluggable external REST client interface.	Live multi-provider LLM integration and empirical benchmarking.
Input Modalities	Plain-text natural-language prompts supported.	Optical Character Recognition (OCR) pipeline for PDFs, scans, and images.

International Journal of Innovative Research of Science, Engineering and Technology (IJIRSET)

Impact Factor 9.935

|www.ijirset.com |A Monthly, Peer Reviewed & Refereed Journal| e-ISSN: 2319-8753| p-ISSN: 2347-6710|

Volume 15, Issue 11, November 2026

|DOI: 10.15680/IJIRSET.2026.1511045|

VI. SECURITY MODEL AND TRUST BOUNDARY

Privacy Shield partitions the execution environment into a Trusted Application Boundary (comprising the user's browser session, the Java Spring Boot backend, the internal Python FastAPI microservice, and the token-mapping store) and an Untrusted External Inference Zone (comprising third-party LLM APIs and transit networks outside organizational control).

Within this trust architecture, five core security invariants govern system behavior:
1) Strict Outbound Isolation: Raw token-mapping tables never leave the trusted backend. Outbound HTTP requests to external LLM providers must contain only the sanitized prompt text and non-identifying system instructions.
2) Session-Scoped Placeholder Unpredictability: Placeholders are bound to a unique request or session identifier. Scoping mappings strictly to the originating session prevents cross-user or cross-session token collisions.
3) Response Placeholder Validation: Because external LLMs are untrusted text generators, model outputs may occasionally alter placeholder formatting (e.g., dropping angle brackets), hallucinate unseen tokens, or echo maliciously crafted strings. The Spring Boot validator performs strict allow-list matching against active session keys before executing any substitution.
4) Data Minimization in Operational Logs: Application and error logs are configured to record operational telemetry (such as entity counts and latency) while avoiding storage of raw user prompts or plaintext token mappings.
5) Residual Risk Transparency: Automated PII detection is probabilistic. Quasi-identifiers (e.g., rare job titles combined with dates) or domain-specific secrets not covered by configured NER/Regex rules may remain in the sanitized prompt. Users and administrators must be informed that sanitization reduces rather than eliminates disclosure risk.

TABLE.IV. PRIVACY SHIELD SECURITY MODEL, THREAT VECTORS, AND MITIGATION CONTROLS

Threat Vector / Asset	Trust Boundary Zone	Security Control / Design Mechanism	Implementation & Verification Note
Raw PII Exposure to External LLM	Trusted → Untrusted Boundary	Pre-inference detection and substitution with typed placeholders (<TYPE_N>).	Enforced in middleware; bounded by Presidio recall.
Token Store Unauthorized Access	Trusted Backend Zone	Access restricted to Spring Boot orchestrator; no external network exposure.	Prototype local store; encryption & RBAC planned.
Cross-Session Token Leakage	Trusted Backend Zone	Mappings indexed by unique session/request IDs with lifecycle cleanup.	Requires TTL eviction testing under concurrent load.
Malformed / Hallucinated LLM Tokens	Untrusted → Trusted Boundary	Allow-list validation of token syntax and active session ownership before restore.	Prevents substitution of arbitrary bracketed model text.
Sensitive Data in Application Logs	Trusted Backend Zone	Structured metadata logging excluding raw prompts and plaintext PII maps.	Requires log-auditing check before production release.

VII. COMPARATIVE ANALYSIS

To contextualize the architectural trade-offs of Privacy Shield, Table V compares four representative paradigms for interacting with LLMs when prompts contain sensitive information: (1) Direct Unsanitized LLM Usage, (2) Manual User Redaction, (3) Irreversible Static Anonymization, and (4) Privacy Shield's Reversible Middleware Architecture.

International Journal of Innovative Research of Science, Engineering and Technology (IJIRSET)

Impact Factor 9.935

|www.ijirset.com |A Monthly, Peer Reviewed & Refereed Journal| e-ISSN: 2319-8753| p-ISSN: 2347-6710|

Volume 15, Issue 11, November 2026

|DOI: 10.15680/IJIRSET.2026.1511045|

TABLE V. COMPARATIVE ANALYSIS OF LLM PROMPT PRIVACY APPROACHES

Evaluation Criterion	Direct Unsanitized LLM Prompting	Manual User Redaction	Irreversible Static Anonymization	Privacy Shield (Proposed Middleware)
PII Exposure to External Provider	High (raw prompt sent externally).	Moderate (depends on user vigilance).	Low (detected entities removed).	Low for covered entities (tokens only).
Semantic & Coreference Preservation	Complete (original context intact).	Variable (may hurt readability).	Low (generic redaction breaks links).	High (indexed tokens keep context).
Reversible Output Restoration	Not applicable (raw values exposed).	Manual post-editing by user.	Not supported (values discarded).	Supported via token store & validation.
Automation & Consistency	Automated, zero privacy control.	Manual, slow, inconsistent.	Automated via rule/NER pipelines.	Automated multi-tier REST pipeline.
Latency & Operational Overhead	Minimal (direct API call).	High human effort; zero compute.	Low-to-moderate local NLP cost.	Moderate (REST hop + Presidio + store).

VIII. EVALUATION METHODOLOGY

Because systematic empirical benchmarking has not yet been completed on the prototype implementation, no experimental accuracy percentages, latency measurements, or synthetic performance claims are reported in this manuscript. Instead, this section defines a rigorous, reproducible evaluation protocol to be executed prior to production deployment.

The proposed evaluation will utilize a curated benchmark dataset comprising synthetic and anonymized prompts spanning multiple domains (e.g., academic advising, customer support transcripts, healthcare administrative notes, and financial summaries) annotated with ground-truth entity spans across standard PII categories (PERSON, EMAIL_ADDRESS, PHONE_NUMBER, LOCATION, ORGANIZATION, and structured identifiers). Evaluation will be conducted across six quantitative and qualitative dimensions, summarized in Table VI:

1) Entity Detection Quality (Precision, Recall, and F1-Score): Computed per PII category and macro-averaged across the test corpus under both strict boundary matching (exact character start and end offsets) and partial/overlap matching (character Intersection-over-Union >= 0.5) to quantify boundary drift.
2) Outbound Sensitive-Data Leakage Rate: Automated payload inspection of serialized HTTP requests leaving the Spring Boot orchestrator to verify zero occurrence of ground-truth sensitive substrings and measure the residual leakage rate caused by false negatives.
3) Tokenization and Restoration Robustness: Measured via placeholder survival rate (fraction of placeholders preserved intact by the external LLM) and end-to-end restoration accuracy (fraction of restored responses matching expected ground-truth values without malformed-token corruption).
4) End-to-End Processing Latency Overhead: Measured over repeated trials (reporting mean, median, and 95th-percentile latency in milliseconds) to isolate the time incurred by Spring Boot orchestration, FastAPI/Presidio analysis, and token mapping relative to baseline direct LLM inference.
5) Downstream Task Utility Preservation: Comparative assessment of LLM response quality on sanitized prompts versus unredacted baseline prompts using task-specific metrics (e.g., summarization ROUGE/BERTScore and blinded human rubric scoring for factual completeness and coherence).
6) Security and Trust-Boundary Verification: Verification tests validating session isolation (confirming Session A cannot restore Session B tokens), injection resilience against malformed placeholders in LLM responses, and absence of plaintext PII in application log files.

International Journal of Innovative Research of Science, Engineering and Technology (IJIRSET)

Impact Factor 9.935

|www.ijirset.com |A Monthly, Peer Reviewed & Refereed Journal| e-ISSN: 2319-8753| p-ISSN: 2347-6710|

Volume 15, Issue 11, November 2026

|DOI: 10.15680/IJIRSET.2026.1511045|

TABLE.VI. PROPOSED EXPERIMENTAL EVALUATION DIMENSIONS AND MEASUREMENT PROTOCOL

Evaluation Dimension	Target Metrics	Measurement Protocol & Tooling	Status
PII Detection Accuracy	Precision, Recall, F1-Score (per type & macro)	Evaluate Presidio output vs. ground-truth spans (strict & partial match).	Protocol defined; pending
Outbound Leakage Prevention	Residual Leakage Rate (%), False Negatives	Inspect serialized outbound payloads for exact/fuzzy ground-truth PII.	Protocol defined; pending
Tokenization & Restoration	Placeholder Survival (%), Restoration Accuracy (%)	Check placeholder preservation in LLM output & validate restoration.	Protocol defined; pending
System Latency Overhead	Mean, Median (p50), and Tail (p95) Latency (ms)	Instrument timestamps across tiers over N >= 100 benchmark runs.	Protocol defined; pending
Response Utility Retention	Semantic Similarity & Blinded Human Score (1–5)	Compare sanitized vs. baseline LLM responses on identical tasks.	Protocol defined; pending
Security & Boundary Integrity	Session Isolation Pass/Fail, Log Audit Pass/Fail	Test cross-session replay, malformed token injection, and log files.	Protocol defined; pending

IX. CHALLENGES AND FUTURE SCOPE

Several technical challenges bound the current design and motivate ongoing engineering work:

- **Contextual and Quasi-Identifier Leakage:** Standard NER and Regex recognizers target explicit identifiers but cannot easily detect indirect contextual disclosures (e.g., unique combinations of dates, roles, and events that identify an individual). Incorporating contextual risk scoring and utility-aware sanitization thresholds [2] represents an important extension.
- **Multilingual and Code-Mixed Prompts:** Users in multilingual regions frequently mix English with regional languages (such as Hindi or Marathi) within a single prompt. Extending Presidio with multilingual transformer NER models and region-specific identifier recognizers (e.g., Aadhaar or PAN formats) is planned.
- **Placeholder Drift in Generative Outputs:** External LLMs may occasionally inflect, pluralize, or reformat placeholders (e.g., transforming <PERSON_1> into PERSON_1 or <PERSON_1>'s), complicating exact string restoration. Prompt-conditioning strategies and morphological-aware placeholder parsers are needed to improve restoration resilience.
- **Multimodal Document and Image Support:** The current prototype processes plain-text prompts only. Integrating an Optical Character Recognition (OCR) preprocessing stage will enable sanitization of uploaded PDFs, screenshots, and scanned forms prior to LLM ingestion.
- **Hardened Vault Persistence and Policy Governance:** Future iterations will replace prototype in-memory mappings with hardware-backed or AES-256 encrypted persistent token stores featuring configurable Time-To-Live (TTL) eviction, role-based access control, and formal compliance auditing.

X. CONCLUSION

This paper presented the architectural design, processing methodology, security model, and evaluation framework of Privacy Shield, a privacy-preserving middleware platform for sensitive prompt sanitization in external LLM interactions. By decoupling user-facing presentation in React.js and request orchestration in Java Spring Boot from Python-based FastAPI/Microsoft Presidio PII analysis, the system enables hybrid NER and Regex entity detection, typed placeholder substitution, and validated response restoration within a clearly defined trust boundary. Unlike

International Journal of Innovative Research of Science, Engineering and Technology (IJIRSET)

Impact Factor 9.935

|www.ijirset.com |A Monthly, Peer Reviewed & Refereed Journal| e-ISSN: 2319-8753| p-ISSN: 2347-6710|

Volume 15, Issue 11, November 2026

|DOI: 10.15680/IJIRSET.2026.1511045|

irreversible redaction, reversible placeholder mapping preserves referential structure so external models can generate coherent responses without receiving raw PII. While core orchestration and detection workflows have been prototyped, systematic empirical benchmarking, live multi-provider LLM validation, encrypted vault hardening, and OCR support remain essential next steps. Executing the proposed evaluation protocol will provide the quantitative evidence required to validate detection accuracy, leakage reduction, and latency trade-offs in real-world deployments.

REFERENCES

[1] K. Edemacu and X. Wu, "Privacy preserving prompt engineering: A survey," ACM Computing Surveys, vol. 57, no. 10, 2025. [Bibliographic note: Verify exact article number, page range, and DOI from the ACM Digital Library prior to final camera-ready submission.]

[2] C. Shen et al., "ProSan: Utility-based prompt privacy sanitizer," IEEE Transactions on Information Forensics and Security, vol. 21, pp. 1198–1212, 2026. [Bibliographic note: Verify complete author list and DOI from IEEE Xplore prior to final camera-ready submission.]

[3] Microsoft Corporation, "Microsoft Presidio: Data protection and de-identification SDK," GitHub / Official Documentation. [Online]. Available: https://microsoft.github.io/presidio/ (Accessed: Nov. 2026).

[4] Protect AI, "LLM Guard: The security toolkit for LLM interactions — Anonymize and Deanonymize scanners," Official Documentation. [Online]. Available: https://llm-guard.com/ (Accessed: Nov. 2026).

[5] Association for Computational Linguistics, "Proceedings of the Sixth Workshop on Privacy in Natural Language Processing (PrivateNLP 2025)," ACL Anthology, 2025. [Bibliographic note: Replace or supplement with the specific cited PP-TS paper authors, title, and page numbers once verified from the workshop proceedings.]



INTERNATIONAL STANDARD SERIAL NUMBER INDIA

INNO SPACE SJIF Scientific Journal Impact Factor

doi crossref

निस्केयर NISCAIR

INTERNATIONAL JOURNAL OF INNOVATIVE RESEARCH

IN SCIENCE | ENGINEERING | TECHNOLOGY

9940 572 462 6381 907 438 ijirset@gmail.com

www.ijirset.com

Scan to save the contact details